NEMESIS: JUDGING THE EFFICACY OF OS FINGERPRINTING SYSTEMS

A Thesis

by

MATTHEW JAMES WIECEK

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Dmitri Loguinov
Committee Members,	Riccardo Bettati
	A. L. Narasimha Reddy
Head of Department,	Dilma Da Silva

August 2019

Major Subject: Computer Science

ABSTRACT

The internet has gone from being a small network of niche uses and mostly academic interest to being a vital, foundational, piece of modern infrastructure that the world depends on. Because of its importance, the internet has also become a target and a gateway for malicious entities. Indeed, it has spawned a whole new military dimension: cyberwarefare. Of vital importance to both attackers and defenders is the identification of vulnerable systems connected to the internet.

OS Fingerprinting is one mechanism by which vulnerable systems may be detected. Despite the importance and proliferation of OS Fingerprinting tools, there has not been a systematic effort to evaluate the effectiveness of such tools. We propose dimensionality as a metric for evaluating OS Fingerprinting Systems and provide a Framework to calculate this value. In addition, we identify NMAP as being the premier OS Fingerprinting tool used today and apply our Framework to this tool under various distortions to ascertain its performance based on our metric of dimensionality.

Under it's default conditions NMAP struggles with Firewalls, which are abundant on the internet, and performs poorly. Our Framework can identify confounding signatures within the database which disproportionately harm NMAPs dimensionality and can remove them from the database. Further, we find that NMAP has modest struggles with network jitter, potentially even on local networks. This, combined with NMAPs difficulty with Firewalls suggests that it is ill suited to the task of Fingerprinting Operating Systems over the internet. Lastly, we identify which features are crucial to NMAPs ability to identify Operating Systems. This, in addition to our other findings, potentially points in the directions for improvements to NMAP, both in it's ability to identify an OS over the internet, and in reducing the number of probes needed to do so.

DEDICATION

To Her Majesty the Queen in Right of Canada, long may she reign. To my dear friends who kept me sane. And to my family, who was always there for me.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Dmitri Loguinov as Chair and Professor Riccardo Bettati of the Department of Computer Science and Professor A. L. Narasimha Reddy of the Department of Electrical Engineering as Committee Members.

A partially complete version of zSignature was provided by Zain Shamsi. Further, zScan code used to run an internet wide firewall probe was also provided by Zain Shamsi with the networking driver written by Yue Zhuo; though code used to parse and interpret the results was conducted by the student.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was primarily self-funded with some departmental funds provided in the form of a TA position for two semesters.

Computing Resources

Portions of this research were conducted with high performance research computing resources provided by Texas A&M University (`https://hprc.tamu.edu`).

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The internet has grown from being a small experimental computer network to being a vital piece of infrastructure for the modern world. Its importance is on par with that of electricity and arguably more important than the telephone or postal system. As the internet has grown in importance, so has the incentive to exploit vulnerable devices on the internet. By finding vulnerable systems, hackers can pilfer the financial information of 10s-100s of millions of individuals [1], cripple hospital systems with randsomware [2], and create botnets spanning millions of computers [3]. Generally, the responsibility for securing systems falls to the owners of the physical machines, regardless of how well qualified or equipped those owners are. Given that most users of internet-connected devices are not security experts, there is considerable concern surrounding the general cybersecurity readiness of the internet at large. Indeed, the Government of Japan considers the situation so dire that the Japanese *Ministry of Internal Affairs and Communications* will be conducting a mass hack of vulnerable Internet of Things (IoT) devices within it's borders in an effort to secure them ahead of the 2020 Olympics in Tokyo [4].

Both attackers and defenders need tools to categorize and identify vulnerable machines operating on the internet. Defenders may wish to do a census of devices on the internet to gauge the scale of vulnerable devices and to identify them for possible remediation. Attackers also wish to do a census of vulnerable devices to find targets for exploits. One key piece of knowledge about said machines is what Operating System (OS) it is running. The vulnerabilities of a Windows desktop machine are generally different from those of a smart fridge. The process of identifying the OS of a remote machine is called Operating System Fingerprinting. There are quite a few OS Fingerprinting tools available today, including Ettercap [5], p0f [6], and NMap [7]; though both Ettercap and p0f are only capable of identifying the OS of systems running on a local network and not the internet at large. Indeed, identifying Operating Systems on a local network is considerably easier than on the open internet. If the devices are on the same physical link layer then the MAC address is exposed and identifying the device may be as simple as looking up the network card

manufacturer by MAC address; this is especially true if the device is an IoT or smartphone device with a custom network card. Even if the MAC address is not visible, devices on LAN tend to have no serious security between them: probes tend not to be blocked, network congestion tends to be minimal, etc. And if all else fails, one can simply walk over to said LAN device and identify it by visual inspection. Meanwhile, almost all internet connected devices can be expected to have firewalls in place, MAC addresses will be unavailable, network congestion unpredictable. Identifying the OS of an internet connected device is considerably more difficult, not least because visual inspection of said device is generally not possible, let alone practical.

Despite the difficulty of running OS Fingerprinting on the open internet, there does exist a standard widely used tool to do so: Network MAPper (NMAP) [7]. Indeed, the NMAP OS Classifier is the de-facto standard for OS classification; it essentially serves as the ground truth for most OS classification tasks. Large internet fingerprinting scans typically either rely on NMAP directly [8] or on connection-less OS Fingerprinting which gauges its effectiveness against NMAP [9]. Indeed, so undisputed is NMAPs reign, that almost any paper touching on OS Fingerprinting or Honeypotting must, at the very least, mention it [10, 11, 12, 13]. Further, when systems are designed to defend against OS Fingerprinting, their effectiveness against NMAP is mentioned [14, 15]. That being said, despite NMAPs reign, there is relatively limited literature regarding the effectiveness and correctness of NMAP.

We propose the Nemesis Framework to evaluate the effectiveness of OS Fingerprinting tools which are intended for internet wide OS Scans. Our key metric of effectiveness will be the dimensionality of the fingerprinting database. A low dimensionality implies that few devices can be reasonably differentiated, whilst a high dimensionality implies that many devices can be reasonably differentiated; the higher the dimensionality, the more effective the classifier. The Framework is capable of simulating three main types of distortions: firewalls, network jitter, and user modification of features. Firewalls are relatively common on the internet and many systems come with them enabled by default. They manifest by a total blocking of a probe. Network jitter can subtly distort probe features that depend on the Round Trip Time (RTT) of the probe. The internet

operates on a “best effort” principle and transit times will vary depending on network congestion, network routing, OS load, et cetera, which inevitably result in variations in the RTT of a packet. Finally, users may sometimes tweak the parameters of their network stack, either in an attempt to improve performance, or to intentionally make their system more difficult to fingerprint. While our dimensionality calculator is, generally speaking, fingerprinter agnostic, we will be targeting NMAP as a practical matter; both to demonstrate the effectiveness of our tool and because NMAP is the de-facto OS Fingerprinting standard.

2. UNDERSTANDING NMAP

NMap is a large tool that does more than just OS Fingerprinting. It's main overall goal is vulnerability detection, of which Fingerprinting is just a small part. In the context of this work, only the components directly related to OS Fingerprinting are relevant. Therefore, we will restrict our discussion of NMAP to the probes it sends when fingerprinting an OS as well as the classifier itself.

When NMAP operates in OS classifier mode, it generally first does a port scan to find one open TCP port and one closed TCP port. This is actually a fairly intensive scan, but can be avoided by specifying which ports to scan. Thereafter, NMAP sends 16 unique probes out, which are listed in Table 2.1 and further described in Section 2.1. The responses to these probes are then placed into a NMAP signature struct. This struct does, in fact, have a human readable version, an example is displayed in Figure 2.1. Please note that this is a signature from the NMAP database, a measurement obtained from a scan does not have ranges for values. So for "SP=FC-106" during an NMAP scan of said system, the result would be a single hexadecimal number, hopefully within the expected range. To understand the meaning of all of the features (such as SP, DF, etc.) it is recommended that one read the NMAP documentation on the matter located at [16]. However, a brief description is also available in Appendix A.1. Once NMAP receives responses to its probes, or the probes time out, it passes the results struct to the NMAP OS Classifier.

Probenum	Explanation
1-6	Valid TCP SYN packets to open port with various options
7-9	Various invalid TCP packets to open ports
10-12	Various TCP packets to closed ports
13	Valid TCP SYN packet to open port with ECN flag set
14-15	ICMP Echo packets
16	UDP packet to closed port

Table 2.1: Various NMAP probes and a brief description of the packet sent

```
Fingerprint (3398) Microsoft Windows 10
Class Microsoft | Windows | 10 | general purpose
CPE cpe:/o:microsoft:windows_10 auto
SEQ(SP=FC-106%GCD=1-6%ISR=108-112%TI=I%CI=I%II=I%SS=S%TS=A)
OPS(O1=M4ECNW8ST11%O2=M4ECNW8ST11%O3=M4ECNW8NNT11%[line truncated]
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=7B-85%TG=80%W=2000%O=M4ECNW8NNS%CC=N%Q=)
T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=7B-85%TG=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=7B-85%TG=80%CD=Z)
```

Figure 2.1: What a Windows Signature looks like in the NMAP database

The NMAP OS Classifier is rather simple. The classifier looks through each category in the DB entry (reference signature) and compares it against the same category in the captured measurement. Each individual test within the category, for example the Window (W) test, is then compared against the measurement. If it is a match, NMAP increments a "NumMatchPoint" counter. The amount of points incremented is governed by the MatchPoints structure defined in the NMAP database. This structure is global for all entries in the database.

Once all tests and categories of tests are compared, "NumMatchPoint" is divided by the amount of "PossiblePoints." Note that if a test, like the Window (W) test, is not in the reference signature or the measurement, the test is skipped and the amount of points assigned to that test by the MatchPoints structure is not included in the number of "PossiblePoints." In effect, not including a test, like the Window (W) test, turns that test into a wildcard. Thus, reference signatures with few tests will match against most measurements and will pollute the results of the NMAP OS Classifier.

2.1 The NMAP Probes

Note that for probes 7-12 the probes have TCP Options set in the following order: Window Scale (10), NOP, MSS (265), Timestamp, SACK permitted. Expected responses to probes 1-6 and 13 are TCP packets with the SYN and ACK flags set. Expected responses to probes 7-12 are TCP packets with the RST flag set as those probes are not valid packets in that context. Expected response to probes 14 and 15 are ICMP ECHO Replies. Expected response to probe 16 is an ICMP Port Unreachable message.

1-6. A valid TCP packet with the SYN flag set sent to an open port. The next 5 packets are the same. All of them have the Timestamp and MSS TCP Options. The order of TCP Options and number of NOP options vary, as do the Window and MSS values. The first probe is used to set the parameters for the T1 test, whereas all the probes (1-6) are used to set the SEQ (TCP ISN Generation tests), OPS (TCP Options), and WIN (TCP Window Size) tests.

7. A TCP packet with no flags set sent to an open port. This is for the T2 test.

8. A TCP Rainbow packet sent to an open port, rainbow meaning that several conflicting TCP flags are set. In this instance, the TCP flags set are SYN, FIN, URG, PSH flags. This is for the T3 test.

9. A TCP packet with the ACK flag set sent to an open port. This is for the T4 test.

10. A TCP packet with the SYN flag set sent to a closed port. This is for the T5 test.

11. A TCP packet with the ACK flag set sent to a closed port. This is for the T6 test.

12. A TCP Rainbow packet sent to a closed port. In this instance, the TCP flags set are FIN, PSH, URG. This is for the T7 test.

13. A valid TCP packet with the SYN and ECN flags set sent to an open port. The ECN flag is somewhat escoteric, it's for Explicit Congestion Notification. The receiving system is free

to ignore the ECN flag if it does not support it. This is for the ECN test and should be looked at in the same context as T1 test.

14. An ICMP Echo Request with the Code field set to 9. Note that according to the standard [17] an ICMP Echo Request must have it's Code field set to 0, making this a malformed ICMP Echo Request. That being said, most Operating Systems simply do not check the Code field on an ICMP Echo Request. Both this probe and probe 15 must return to set values in the IE test.

15. A valid ICMP Echo Request, the Code field is set to 0 this time. Both this probe and probe 14 must return to set values in the IE test.

16. A UDP packet with ascii character "C" repeated 300 times in the data field. Sent to a closed port. This sets U1 field.

2.2 Formal Definition

Assume NMAP is attempting to compare a measurement to a particular reference signature in the NMAP database. We define $R_i = 1$ if received response to probe i. $R_i = 0$ otherwise. We also define $R'_i = 1$ if db signature expects response to probe i. $R'_i = 0$ otherwise. Basically, we're just defining whether the probe came back and whether or not it was supposed to come back.

NMAP cares not only about whether probes came back, but what information it can obtain based on those probes coming back. These discrete pieces of information are called features; for example, what TCP Options were enabled on the target response to our probe. We define $F_{ij} =$ feature j in response to probe j. We further define $F'_{ij} =$ feature j db signature expects in response to probe j. Note that because these features must be derived from responses to NMAP probes we have $R_i = 0 \implies F = \{F_{ij} = \emptyset : \forall j \in F_i\}$ as well as $R'_i = 0 \implies F' = \{F'_{ij} = \emptyset : \forall j \in F'_i\}$.

Sometimes NMAP cannot derive a feature based on the probe responses it receives, other times the database doesn't define what value a received feature is supposed to have. There is no penalty in either of those cases in the event of feature mismatch. Therefore define indicator variable $Z_{ij} = 1$ if $F_{ij} \neq \emptyset$ and $F'_{ij} \neq \emptyset$. $Z_{ij} = 0$ otherwise.

NMAP attempts to Fingerprint an OS both based on which probes it responds to and how it responds to those probes. It defines a global weighting both for whether the probe response pattern matches, as well as whether the features derived from the probe responses match what is in the NMAP DB. We define w_i as the weight assigned to whether R_i matches R'_i. We also define w_{ij} as the weight assigned to whether F_{ij} matches F'_{ij}.

So far, everything is reasonably straightforward. However, NMAP defines certain conditions that warrant special exceptions in an attempt not to unfairly penalize a system for not matching. For example, certain probes must be sent to an open port. If NMAP cannot find an open port it will not send certain probes and, therefore, won't include the weighting for a mismatch. Thus we define indicator variable $Y_i = 0$ for $i \in 1, 7, 8, 9, 13$ if no open port found and $Y_i = 1$ for $i \in 1, 7, 8, 9, 13$ if an open port is found.

Note that indicator $Y_i = 0$ for $i \in [2, 6]$ as NMAP assumes that a target system will treat the first 6 probes the same. We also have $Y_i = \neg R'_i$ for $i \in [14, 16]$ as NMAP only assigns probe response weight to the ICMP and UDP probe responses if and only if it doesn't not expect these probes to return. Thus a target system may be penalized if these probes return, but NMAP wasn't expecting it to. If NMAP was expecting these probes to return and they do not, there is no penalty. $Y_i = R_i$ otherwise as for the remaining probes, the various probes sent to closed ports, there is no penalty if they do not return. It is not always clear whether or not all the ways that indicator variable Y_i is set is intentional.

We thus have NMAPs confidence score calculated as:

$$\frac{\sum_i (Y_i 1_{R_i = R'_i} w_i + \sum_j (Z_{ij} 1_{F_{ij} = F'_{ij}} w_{ij}))}{\sum_i (Y_i w_i + \sum_j (Z_{ij} w_{ij}))} \tag{2.1}$$

2.3 Examples

2.3.1 More Features, More Problems

The way the NMAP classifier works can lead to unintuitive results. For instance, in certain cases, adding measured features can reduce the effectiveness of NMAP in correctly identifying an

operating system. Consider the following reduced example where we are scanning a hypothetical system “X.” There are exactly 2 signatures in the database “X” and “Y.” The reference signatures are reproduced below:

X	Y
OPS(O1 = M500)	OPS()
WIN(W1 = 200)	WIN()
T1(TG = FF)	T1(TG = FF)

Now consider the following situation with measurement:

OPS(O1 = M500)

WIN(W1 = 200)

T1()

In this circumstance, the measurement will perfectly match “X” with a confidence score of 1.0 as it’s supposed to. However, when we add in feature TG we have measured signature:

OPS(O1 = M500)

WIN(W1 = 200)

T1(TG = FF)

Now, this signature matches with confidence score 1.0 for both “X” and “Y.” We can no longer differentiate between the two entries in the database. Adding a feature to the measurement reduced the effectiveness of the classifier.

Name	Windows	Tomato	SYN Only Windows	Match Points
SP	FC-106	∅	FF	25
GCD	1-6	∅	1	75
ISR	108-112	∅	110	25
TI	I	∅	I	100
TS	A	∅	A	100
O1..O5	M4ECNW8ST11	∅	M4ECNW8ST11	100
O6	M4ECST11	∅	M4ECST11	20
W1..W6	2000	∅	2000	90
DF	Y	Y	Y	20
TG	80	40	80	15
S	O	O	O	20
A	S+	S+	S+	20
F	AS	AS	AS	30
RD	0	0	0	20
Q	""	""	""	20
R1..R16	1111111111111111	1000000001100000	1111110000000000	540

Table 2.2: Comparison of Windows and Tomato Signatures against the same Windows machine when behind a SYN-Only Firewall.

2.3.2 Tomato Router

The Tomato router happens to be an incredibly confounding signature when faced with a SYN only firewall. We can demonstrate this by calculating the NMAP confidence score for a typical "Windows 10" signature against both itself and against Tomato when faced with a Firewall that only allows valid SYN packets through (without the ECN flag). As many features don't make it back, I will simplify the signatures to only show relevant fields because non-existent features, due to probe non-response, don't carry a penalty. However, for completeness, the full signatures will be reproduce in the appendix: Tomato in Appendix A.2 and Windows 10 in Appendix A.3. I also construct a signature from the Windows signature under the assumption that it is blocked by Firewalls. For all probes that come back, the relevant features remain. For probes that do not make it back, the features are not listed. Where are range exists, a value approximately within the center of the range is picked.

When looking at Table 2.2 a few things are immediately striking. First, Tomato has an awful lot of null fields. Looking back at the Formal NMAP definition in Section 2.2 we can see that null

fields essentially act as wild-card fields because there is no penalty for failure to match. We can see that of the fields remaining, the only fields where Windows matches itself but Tomato does not is the "TG" field, which is only worth 15 points for matching. Meanwhile, the Response/Non-Responsiveness to NMAP probes is worth a whopping 540 points. Further, even just eyeballing it, we can see that Tomato is closer to the simulated SYN-Only Windows measurement than the actual Windows signature simply because Tomato matches the probe response pattern that Windows exhibits when it's behind a Firewall.

When actually using the Formal NMAP definition to calculate confidence score for matches we get a confidence score of 0.8321 for Tomato but only 0.6393 for Windows. This despite the fact that we are still scanning the same Windows machine, simply behind a Firewall. This confounding effect is actually quite severe for the Tomato signature and may, in fact, be severely polluting the results of actual NMAP scans on the open internet. For example, during an internet wide OS Fingerprinting scan using NMAP, [18] found that 21% of all hosts scanned had Tomato appear as a potential OS candidate. Given the relative obscurity of the Tomato OS, and the above example, it is clear that confounding signatures can pose a severe problem for NMAP.

3. NEMESIS DATABASE REDUCTION AND DIMENSIONALITY

Our preferred metric for determining OS Fingerprinting performance is the dimensionality of the underlying database. The higher the performance, the larger the dimensionality. Using this as a metric allows us to evaluate the effects of distortion and confounding signatures on the performance of the database. For example, the Tomato router mentioned in Section 2.3.2 is clearly confounding, but how big of an effect does it have on NMAPs performance? To answer these sorts of questions we propose the Nemesis Database Reduction Framework. This Framework is Fingerprinter agnostic, that is, the technique will work with any underlying OS Fingerprinting system. The only things that are Classifier dependent are the number of signatures in the underlying database, a technique for distorting them (if applicable), and a way to determine what OS the Classifier believes the fingerprint belongs to (whether it self-matches or not). We can then eliminate undesirable signatures until we are left with the reduced database. The remaining number of signatures in the database constitute the dimensionality of said Classifier.

In general, there are two different categories of signature that we may wish to eliminate: confounding signatures and weak signatures. A confounding signature is one that, no matter what signature is simulated, the Classifier will claim it matches against the confounding signature. A weak signature is one where it fails to match against itself when simulated.

When running a DB reduction, a matrix of $num\_signatures \times num\_signatures$ is created with each value being initialized to zero. Each signature in the Classifier database is then simulated and passed into the Classifier. Each signature in the db is labeled as an integer $[0, n)$. Matrix value $(simulated\_signature, top\_match\_signature)$ is then incremented by 1. In the event that there is a tie for the top match, each matrix value with a top match is incremented. If each signature matched itself and only itself we would have a matrix with a perfect series of ones along the diagonal. If a signature is confounding and will match against any simulated signature, it would have a perfect column of ones. If a simulated signature is weak, it will let many signatures match against it and have a row of ones along its row. See figure 3.1.

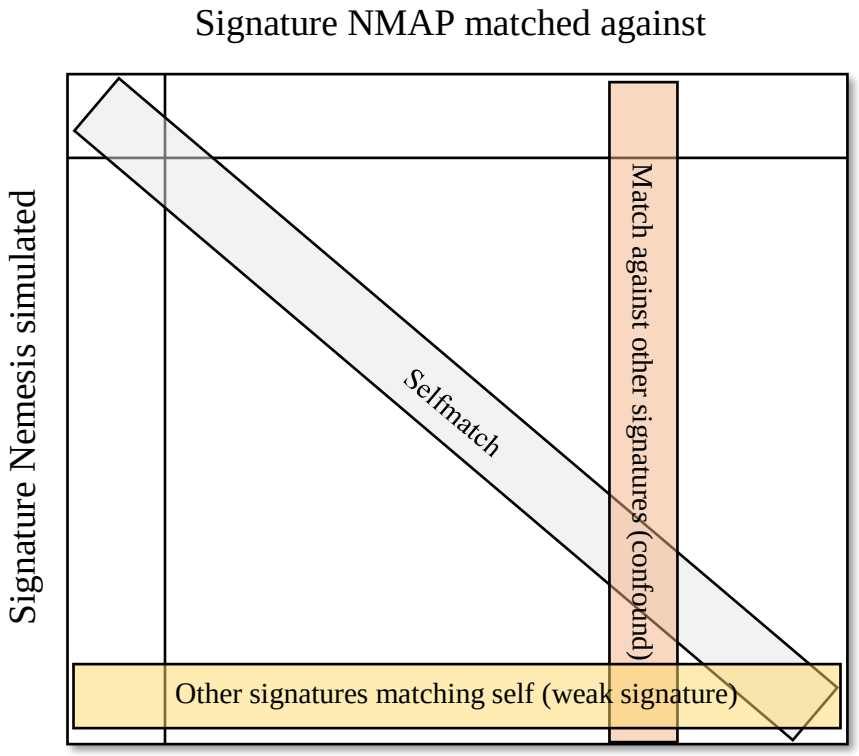


Figure 3.1: Nemesis Reduction Matrix

When reducing, we first drop confounding signatures. The confound score is defined as the following:

$$confound\_score_i = \frac{col\_sum_i - matrix(i,i)}{num\_signatures}$$

After generating the reduction matrix and calculating the confound score for each signature, the signature with the largest confound score is dropped, as long as said signature is above the maximum confound threshold. Once a signature is dropped, a new reduction matrix is generated and the process is repeated until there are no more signatures left above the confound threshold. After dropping the confounding signatures, we move on to dropping weak signatures. The weakness score is 1 if the signature failed to match itself. Otherwise it is calculated as follows:

$$weakness\_score_i = 1.0 - \frac{matrix(i,i)}{row\_sum_i}$$

Like before, we generate the reduction matrix and calculate the weakness score for each signature. If the maximum weakness score is greater than the drop threshold, it is dropped, we generate the new reduction matrix and repeat the process until there are no more signatures above the threshold. The final number of signatures left in the database is the dimensionality of the classifier.

While the above technique does work with any classifier, it does require that certain classifier specific components be built to actually run useful tests. As NMAP is the de-facto standard for OS Fingerprinting, we choose to build said components to target it and demonstrate the usefulness of our dimensionality reduction technique.

4. NEMESIS NMAP CORE

As mentioned in Section 3, the Nemesis Reduction tool depends on a Classifier specific implementation that can feed it certain data and perform various distortions. The Nemesis NMAP Core handles most of the NMAP specific tasks like generating valid signatures from the database, distorting said signatures, and calculating NMAP confidence scores.

The NMAP Core is built around actual NMAP code from the NMAP Open Source repository. The NMAP OS Classifier, NMAP DB Parser, and their dependencies were isolated from the full source code, and the NMAP Core was built around them. Generating valid signatures with no distortion is relatively straightforward. For fixed field features, the value is simply copied; if there are multiple values with an OR, the first value is chosen. In the event that the feature is a range, then the midpoint of the range is chosen. That being said, there is a catch. Certain fields are only available under certain circumstances. The TG feature (Time to Live Guess) is only available if the UDP probe did not return. Thus, for a "no distortion" signature this field must be filtered out.

There are three fundamental distortions: probe packet drops due to Firewalls, network jitter distorting values that depend on RTT, and deliberate distortion of OS features. Even the simplest distortion to model, Firewall drops, is somewhat difficult to model. NMAP has 17 well defined probes which are transmitted to the target machine. For any given probe, all the dependent features must be identified and either dropped or "un-dropped" while the probe response flag is set to false. This means that all the fields on the relevant probe line, see Figure 5.1, need to be dropped, save for the probe response field R is set to N. Again, there are a few additional special features located on different signature lines that are nonetheless dependent on multiple types of probes. The way NMAP Core handles this is by first generating a "clean" signature with all features properly simulated and no features pre-maturely dropped. Then, based on user input defining the probability of Firewall drop for each probe, a coin is flipped to determine whether or not a given probe is dropped. Then based on the results, the appropriate features are removed.

4.1 Feature Distortion

4.1.1 Picking Reasonable Values

Feature distortion is considerably more involved than merely dropping probes. In particular, how does one pick reasonable distort values for the features being distorted? There are a few possible distortion strategies.

Define i as the reference signature we wish to distort and j as any other reference signature in the NMAP database.

1. Uniform: For each field we wish to distort in i we uniformly at random choose a valid value in the range.

2. Uniform Plausible: For each field we wish to distort in i we choose a random j and replace the relevant field in i from the same field in j. If the field in j is the same as the one in i, choose a different j.

3. Unstable Patch (naive): For a given i, first choose a random j. Then, for each field we wish to distort, replace the relevant field in i from the one in j.

4. Unstable Patch: For a given i, first choose a random j. Verify that, for each field we wish to distort, j has a different value. If it does not, keep randomly picking different j until this is the case. Then, for each field we wish to distort, replace the relevant field in i from the one in j.

5. Stable Patch: Ahead of time, for every i in the NMAP database, create a mapping to a j where there does not exist any overlap in the fields between i and j. If this is not possible, pick the j with the least overlap with i. When we wish to distort a given i, pick the j based on the established mapping and, for each field we wish to distort, replace the relevant field in i from the one in j.

It's clear from the list above that there are many possible user distortion models. In particular, we chose distortion model 5, stable patch. The reason being that we can reasonably compute

this ahead of time, while also getting good, valid, distort values. The patch generator finds stable mappings for each NMAP signature i.

The basic operation is quite simple. For each signature i in the NMAP database we identify which features it contains. Then we construct a set $candidates$ which contains every signature j in the NMAP database whose features are a superset of the features in i. We then find the signature j that has the least overlap with i; that is, the signature j that i has the lowest match score against. In the event that there is a tie amongst signatures with the lowest match score against i, one of those signatures is chosen at random.

There are, however, practical considerations. There are many fields that have low entropy as shown by Figure 4.1. Therefore there are some fields that are not only unlikely to have a patch candidate, but due to their low entropy, are unlikely to have much useful signal anyway. We thus allow the patch candidate system to exclude fields when searching for candidates. In general, when running the patch system as part of NMAP DB Reduction (see section 3) we exclude all fields not under distortion as we do not need a candidate for them anyway. When running as a standalone application, we tend to simply exclude the low entropy fields (see section 4.1.3): [CC, CD, II, Q, R, RD, RID, RIPCK, RIPL, RUCK, RUD, SS, UN, O, S, A, F].

4.1.2 Running the Distortion

Once we have a patch signature p generated for each signature in the database x, we know have a reasonable set of values to distort the features into. Before running a distortion, we first specify the probability that any particular feature is distorted; For example, the Time-To-Live (T) feature or the Options feature. We group features together by type. So, for instance, if the user sets a 20% probability of distorting the Window size (W), there is one coin flip, with 20% chance of heads, and if the coin is heads all of the W features are distorted. This is because we assume that a feature won't be distorted on a per NMAP probe basis but on a machine wide basis. Thus, all NMAP probes should experience the same distortion. There is also a slight inter-dependency between the TCP Options (O) feature and the Timestamp (TS) feature. In particular, if the TCP Options are distorted such that they do not have a Timestamp as part of the options string, then the

TS feature must be set to 'U' meaning unsupported. This dependency check happens after the coin flips and before the distortion. In addition, the Time-To-Live Guess (TG) feature is considered a Time-To-Live (T) feature; this is because the existence of these two features is mutually exclusive and depends entirely on whether the U1 (UDP probe to closed port) comes back. In the event it does exact TTL calculation may be performed; if not a slightly less precise estimate must take place. In the end, however, both T and TG are attempting to measure the same underlying property of the OS.

4.1.3 Which Fields Are Worth Distorting?

The NMAP database contains many different features. However, it is reasonable to assume that not all of those features carry useful signal when attempting to differentiate amongst various Operating Systems. One possible way to help narrow down which fields provide useful data is by looking at the entropy of the values of the fields in the NMAP database. It is clear from Figure 4.1 that fields SP, ISR, OX, and WX have the highest entropy. Note, however, that ISR and SP are ranges and that Figure 4.1 treats different ranges as different values, even if they overlap. Thus, excluding SP and ISR, it is clear that TCP Options (OX) and TCP Window Size (WX) contain the most amount of entropy in the database and should therefore be able to provide the most amount of differentiability. Note that the number of available features is restricted when firewalls are in existence.

While certain features may have high entropy, they may not convey as much signal as expected due to the way they may correlate with other features. For example, the TS field is dependent on the value in the OX field. If timestamp is not specified as one of the available options, then the TS field must be set to 'U' for unsupported. Thus, to consider how much value each field actually provides, we can look at a the results from a simulation. We first configure a Firewall operating under the SYN-Only case, all other packets have a 100% chance of being dropped. We set the maximum confound (column) threshold to 0.0225 (threshold determined in Section 6.2). After that reduction is complete, we then run a weakness (row) reduction with maximum threshold of 0.20. We can look at how good each DB signatures self-match is when only one particular field is

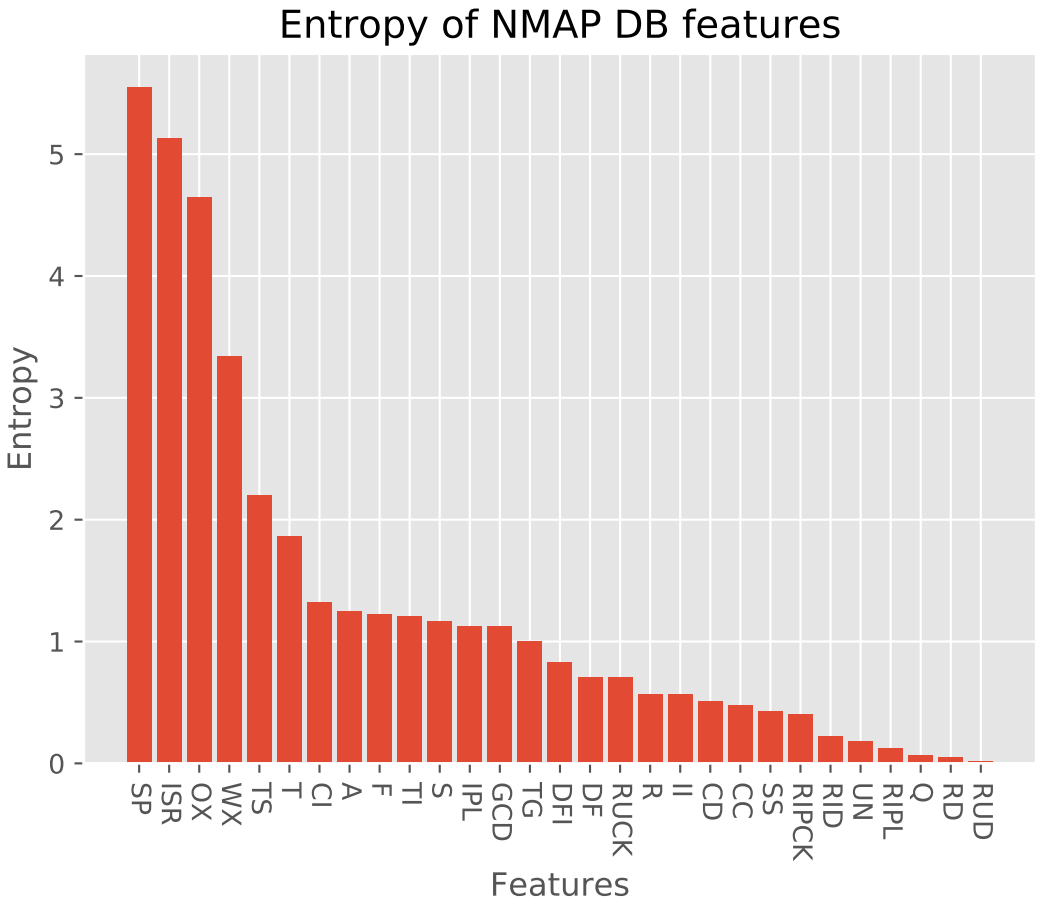


Figure 4.1: NMAP DB Feature Entropy

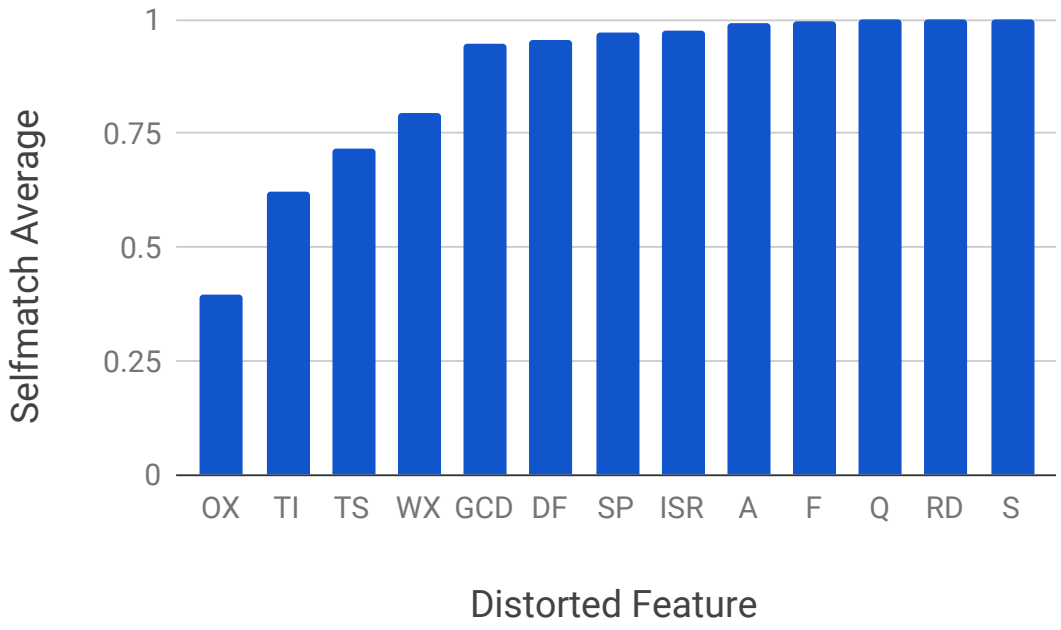


Figure 4.2: Signature selfmatch average when only one feature is distorted in the RAC SYN case with 0.0225 confound maximum. Features that do not affect selfmatch truncated.

distorted. Looking at figure 4.2 it may not come as much of a surprise that distorting TCP Options (OX) carries a strong effect; though it is certainly surprising how important IP ID Generation (TI) is to ensuring a correct self-match. This is almost certainly due to a reasonably heavy weight on 'TI' as well as the fact that certain OS Families have rather distinct 'TI's': MS Windows increments up by 256 in little-endian.

From Section 5 we know that there are several common possible Firewall scenarios. We thus can look at the effect of distorting a given feature in less restrictive environments than in the SYN-only case. As more features survive the Firewall, we would expect to see more features play a role in the Fingerprinting of the OS. As such, we can rerun the above experiment with progressively more permissive Firewalls: consult Appendix B for the Firewall configurations. As before we run the simulations with a confound reduction first, with the maximum confound threshold set to 0.001 this time, followed by a weakness (row) reduction with threshold of 0.20. We look at the signature self-match average when only one field is distorted as before. Looking at Figure 4.3

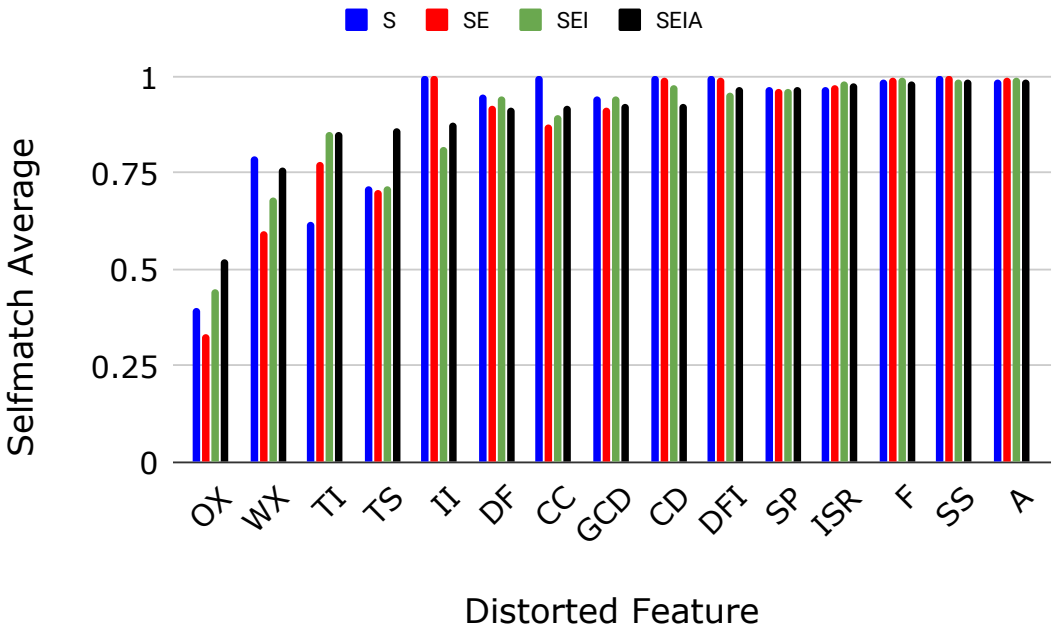


Figure 4.3: Signature selfmatch average when only one feature is distorted under various Firewall scenarios. Features that do not affect selfmatch truncated.

we can see that, as before, the top 4 most import features are the same: TCP Option (OX), TCP Window Size (WX), TCP IP ID Generation (TI), and Timestamp (TS). Note that the relative order of importance for these features various somewhat with the Firewall configuration. Unsurprisingly, a feature like ICMP IP ID Generation (II) only affects signatures when the Firewall actually allows through ICMP packets. Overall, though, the vital features are generally quite similar between all of the different Firewall types and, besides IP ID Generation (TI/II) and the overlapped values (SP, ISR, GCD), they line up reasonably well with entropy in Figure 4.1.

4.2 Jitter

While local networks under low utilization may be expected to have reasonably consistent Round Trip Times (RTT), the internet is under a constant state of flux. The internet operates under a best effort model, and though packets usually make it to their destination, the RTT's of packets tend to have some level of variance. This causes problems for OS Fingerprinting tools that try to derive information about how time-dependant algorithms in the OS Network Stack work. In the

context of NMAP, there are generally two features it watches out for that depend on time. IP ID Generation, and TCP Initial Sequence Number (ISN) generation.

From the NMAP reference guide, for the IP ID tests, there are 6 possible outcomes:

---

1. If all of the ID numbers are zero, the value of the test is Z.

2. If the IP ID sequence ever increases by at least 20,000, the value is RD (random). This result isn't possible for II because there are not enough samples to support it.

3. If all of the IP IDs are identical, the test is set to that value in hex.

4. If any of the differences between two consecutive IDs exceeds 1,000, and is not evenly divisible by 256, the test's value is RI (random positive increments). If the difference is evenly divisible by 256, it must be at least 256,000 to cause this RI result.

5. If all of the differences are divisible by 256 and no greater than 5,120, the test is set to BI (broken increment). This happens on systems like Microsoft Windows where the IP ID is sent in host byte order rather than network byte order. It works fine and isn't any sort of RFC violation, though it does give away host architecture details which can be useful to attackers.

6. If all of the differences are less than ten, the value is I (incremental). We allow difference up to ten here (rather than requiring sequential ordering) because traffic from other hosts can cause sequence gaps.

7. If none of the previous steps identify the generation algorithm, the test is omitted from the fingerprint.

Above list reproduced from [19, Chapter 8].

It is clear from the above, however, that the IP ID Generation feature is reasonably robust. Though it does depend on time, the assumption is that the probes are sufficient close together in

time that the Δ_{Time} between probes can be safely ignored. Indeed, the primary distortion of the IP ID is likely to be how high of a load the target is under as IP IDs must be unique. If the target is under extremely high load, the IP ID will be forced to tick up at an accelerated rate and may force the NMAP OS Classifier to misidentify the IP ID generation algorithm.

Initial Sequence Number (ISN) feature identification, however, is considerably more complicated than the IP ID generation as it relies on the timing between probes; in particular, the Δ_{Time} for TCP Syn-Ack transmissions from the Target back to NMAP. The three fields that are used to quantify the TCP ISN generation:

1. ISR: this quantifies the average rate that the ISN ticks up per unit of time.
2. SP: The variance in the rate that the ISN ticks up per unit of time.
3. GCD: The greatest common denominator of the Δ_{ISN}. That is, the smallest value by which the target increments its ISN value.

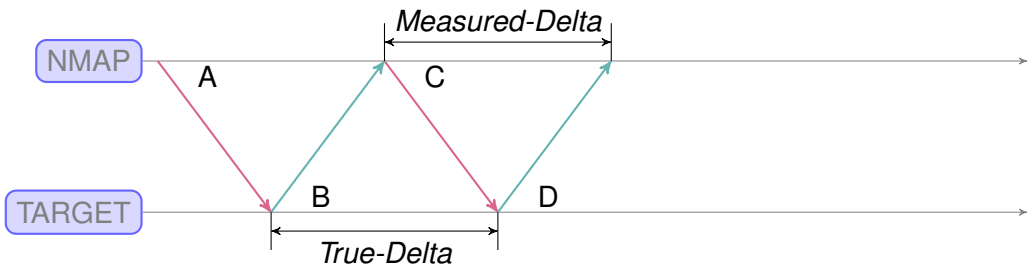


Figure 4.4: NMAP measurements subject to jitter due to difference between True-Delta and Measured-Delta.

Both ISR and SP values usually have a range, whilst the GCD tends to have a list of possible acceptable GCD values. Note how NMAPs measurements of both ISR and SP (being dependent on the time) make said measurements susceptible to network jitter. If we assume that there is zero processing time between the receipt of an NMAP probe and the sending of a response, we can see

that the true Δ_{Time} between SYN-ACK responses is the transit time of $B+C$ (shown in figure 4.4). However, what NMAP is capable of measuring is the Δ_{Time} between $C + D$ (also shown in figure 4.4). This isn't a problem if the transit times don't change between probes, but when network jitter does occur it can throw off NMAP measurements.

4.2.1 Jitter Simulator

In principle, the fundamental idea behind the jitter simulator could be applied to any time sensitive field for any OS Fingerprinting system. In this instance, however, we choose to build a system to apply jitter to the NMAP Initial Sequence Number fields.

Once again, looking back at Figure 4.4 we can see that the problem of jitter occurs because our measurement of Round-Trip-Time (RTT) isn't necessarily the same as the actual RTT. The more "Measured-Delta" diverges from "True-Delta" the worse the effects of network jitter on the feature we are trying to measure. The first problem we run into is that there is no universally agreed upon theoretical method of modeling network jitter [20, 21, 22]. That being said, we do not need a perfect method of modeling network jitter, we only need a reasonable approximation of various levels of jitter that may be experienced: low, medium, and high jitter. Finding a jitter model that can accurately represent the internet as a whole in all circumstances is a bit outside the scope of this work. Therefore, we choose to model one-way delays of a packet using a pareto distributed random variable; reasonable parameters for said random variable where acquired from [23]. We use this random variable to calculate the one way trip time for each NMAP ISN probe.

Given the one-way delay values of the NMAP probe, there is only one thing left to do before we are able to calculate the distorted ISN values. We first must determine what the actual initial sequence number would have been at the time of probe arrival. We can the measure what values NMAP would calculate when it receives the responses to it's probes. There is a slight trick here as SP, ISR, and GCD are encoded in a slightly non-intuitive way (see Appendix C). Fundamentally, though, as mentioned earlier:

1. ISR: this quantifies the average rate that the ISN ticks up per unit of time.

2. SP: The variance in the rate that the ISN ticks up per unit of time.

3. GCD: The greatest common denominator of the Δ_{ISN}. That is, the smallest value by which the target increments its ISN value.

Therefore, to calculate the Δ_{ISN} we simply need to take the average rate that the ISN ticks up per unit time (ISR) and, on an alternating basis, either add or subtract the variance per unit time (SP). Naturally, since both ISR and SP are usually given in ranges, we take their mean when using them in calculations. We should now have two Δ_{ISN}'s. One where we add the variance to the ISR and one where we subtract the variance from the ISR. For each probe we now assign a Δ_{ISN}, alternating between the two possible Δ's. We scale them by multiplying each by the calculated "True-Delta"s. Each "True-Delta" can be calculated by taking the one-way delays and adding them together; that is, calculating the last time since a simulated probe arrived (consult Figure 4.4). We now divide each Δ_{ISN} that each probe has by the GCD, round it to the nearest integer, and then multiply it by the GCD. We check to see if the calculated GCD between Δ_{ISN_i} and it's predecessor $\Delta_{ISN_{i-1}}$ matches the target GCD. If it does not, then increment Δ_{ISN_i} by the target GCD. Keep repearting until the actual GCD between Δ_{ISN_i} and it's predecessor $\Delta_{ISN_{i-1}}$ matches the target GCD. We now have the actual Δ_{ISN}'s our target system would have sent out.

Now we simply need to record what NMAP would have measured. This is relatively simple. Take the Δ_{ISN}'s and use the one-way delay values again to calculate what the "Measured-Delta" (see Figure 4.4) would have been for each probe and use that to calculate the SP, ISR, and GCD. This should be as simple as taking the Δ_{ISN}'s and dividing them by their "Measured-Delta" time and then calculating the mean and the variance. We now have our distorted ISR and SP values.

4.2.2 Effect of Jitter on RTT Sensitive Features

We know that the ISN related fields [SP, ISR, GCD] are susceptible to RTT jitter. While our preferred metric for OS Fingerprinting systems is dimensionality, it is of interest to see how various levels of jitter impact the ISN fields ability to stay within their respective ranges. That is, what level of jitter is required to actually negatively impact those fields. For this we run the Jitter Simulator

Scale	Shape	Iterations	% Passed	E[abs(X-Y)]
N/A	N/A	1000	0.979698	0
1	1000000000	10000	0.971071	1.02E-09
1	2.59	10000	0.707468	0.806213
0.5	1.72	1000	0.570661	1.03003
0.27	1.25	1000	0.457493	1.77216
0.25	1.21	1000	0.446234	1.92068

Table 4.1: How often the [SP, ISR, GCD] fields match after being subjected to varying levels of jitter.

(4.2.1) in a standalone mode; we generate the new [SP, ISR, GCD] values after being subjected to jitter and then compare them against the original ranges to determine whether or not they are still within range.

For the test the Scale and Shape parameters for the Pareto random variable were obtained from [23], with the exception of [N/A, N/A] and [1, 1000000000] (using format [Scale, Shape]) as they were meant to simulate no jitter and extremely low jitter, respectively. The results are evident in Table 4.1. As is evident, even in the no-jitter case there are some fields that cannot match themselves, just over 2%. This is likely because some fields cannot be matched in all scanning circumstances and depend on the frequency of probes. For example, supposing a system that incremented it's ISN by 10000 once every 30 seconds. If all 6 ISN probes are sent within 1 second of each other, the ISN counter will appear fixed. If they are instead sent every 30 seconds, the count is accurate. As such, some signatures can only be matched against when the NMAP OS Fingerprinter is set to extremely low aggressiveness i.e. paranoid mode. Looking at the extremely low jitter case, which is meant to represent a Local Area Network, only 3% of signatures fail to have a match. This is similar to the no jitter case and is likely that these signatures simply have either unmatchable or nearly unmatchable ranges for the ISN generation fields.

Looking at the cases which are supposed to more realistically represent the internet in Table 4.1, it is clear that as the level of jitter increases, the less likely said signature is to successfully discern the ISN generation fields. Indeed, in the particularly severe cases of jitter, the success rate

falls to below 45%. As such, it is clear that jitter does have an effect on NMAP being able to correctly discern ISN fields. That being said, it's possible that these fields have a limited role to play in actually fingerprinting an OS. As such, we will wish to evaluate how distorting these fields will affect the dimensionality of the NMAP OS Fingerprinter.

5. INTERNET WIDE FIREWALL PROBE

It is already clear from Section 2.3.2 that confounding signatures can be a big issue. Ideally we would like to measure the effect that such signatures have on the dimensionality of the NMAP OS Classifier; naturally, that requires a method of identifying such signatures. However, it is also clear from the example in Section 2.3.2 that an awful lot of match points are contained within the probe response pattern, which depends on Firewall parameters. As such, it is important to consider our approach to modeling Firewalls on the open internet. We propose a rather simple model for Firewalls: simply define the probability that any given probe is blocked. This models advantage is in its simplicity, all we need to do is determine how likely an OS Fingerprinting probe is to come back.

As a way to guide the firewall parameters in our Nemesis simulations, we ran a quick internet scan to help provide some rough statistics on which NMAP probes will make it through, and which NMAP probes will be blocked. A full list of publicly routable IPs as of January 30th, 2018 was extracted from BGP tables provided by [24]. At that time there were 2865828579 routable IPv4 addresses on the internet. This list was then shuffled and a single TCP-SYN packet was sent via port 80 to each IP in the list. If a SYN-ACK response was detected, that IP was marked alive. A total of 65930168 signatures were marked live. A random coin flip with a 1% chance of heads was flipped and, if heads, said live IP was hit by an NMAP scan. The single TCP-SYN packet would be transmitted from a randomly selected IP from a pool of about 50 IPs. The NMAP scan would also originate from a randomly selected IP from a different pool of about 50 IPs. The NMAP scan assumed port 80 was open and port 81 was closed. All outgoing and incoming NMAP packets were logged. Post-processing then allowed for the matching of outgoing probes and incoming responses to identify the response rate for each type of NMAP probe across the wider internet. These results can be found in figure 5.2. Notably, almost all valid TCP-SYN packets to an open port received a response, even when the somewhat esoteric ECN flag is set. ICMP echo requests usually elicit a response, even if an invalid code is set (code should always be 0 according to IETF standards [17]).

Label	Explanation
S	Base NMAP probe to verify that port 80 is indeed open.
T1/X1	Valid SYN packet
X#	Valid SYN packets (with varying options and window sizes).
T2	No flags set TCP packet.
T3	All flags set TCP packet (rainbow)
T4	TCP ACK packet to closed port.
T5-T7	TCP SYN packet to closed port with varying options.
ECN	Valid TCP-SYN packet with Explicit Congestion Notification flag set.
IE-9	ICMP Echo Request with invalid code set (Code 9).
IE-0	ICMP Echo Request with valid code set (Code 0).
U1	UDP packet to closed port.

Figure 5.1: A brief explanation of the labels in figure 5.2

A TCP-ACK to a closed port is reasonably likely to elicit a response. Unsolicited UDP packets are moderately unlikely to receive responses, as are TCP SYN packets to closed ports. Invalid packets are unlikely to receive any response.

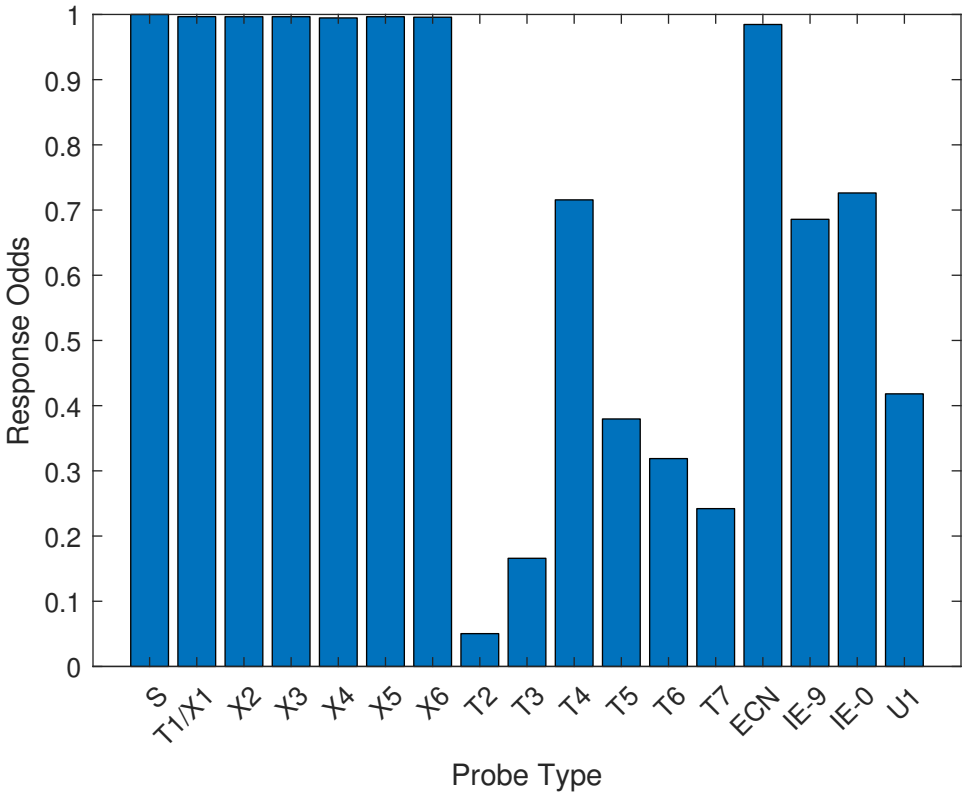


Figure 5.2: Internet Wide NMAP Scan Probe Response Odds

6. CONFOUNDING SIGNATURES AND THEIR EFFECTS

6.1 Visualizing a Confounding Signature

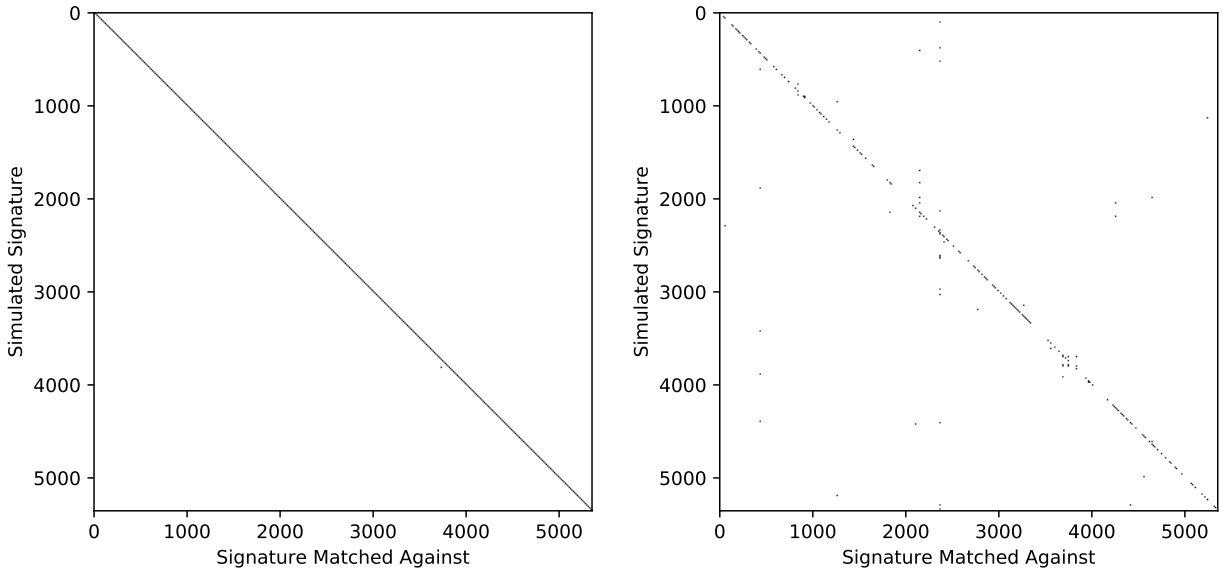
In Chapter 3 we talked about confounding signatures. As a way to concretely visualize the Reduction Matrix, we display an example of said matrix in Figure 6.1. It is clear that in Figure 6.1(a), the no distortion case, most signatures manage to self-match creating a nice clean diagonal line. Meanwhile, when a Firewall is deployed, as in Figure 6.1(b) blocking all packets except for SYN packets (ECN flag allowed), we can see considerably fewer signatures matching as well as evidence of a few confounding signatures (as evidenced by the horizontal lines).

Given the size of the NMAP Database and the resolution of the figures, it is not necessarily obvious how many signatures are confounding. Nor can we reasonably derive information on how these confounding signatures impact the dimensionality of the database.

6.2 How Many Signatures Are Confounding? What Effect Do They Have on Dimensionality?

As mentioned before in Section 3 and shown in Figure 3.1 there are two different ways to run reductions on the database: column-based (eliminate confounding signatures) and row-based (eliminate signatures that fail to self match). We can use the column-based reduction to identify the confounding signatures and remove them. This allows us to not only get an accurate count of the number of confounding signatures, but it will allow us to use the new "clean" database and use it in row-based reductions to determine the effect of confounding signatures on the dimensionality of the database. Basically, we will be able to see count how many signatures will fail to match themselves simply because a confounding signature will match against everything. Before running a column-based reduction, however, we must first define the threshold for what counts as confounding.

To visualize how confounding most signatures are we first run one iteration of Nemesis DB Reduction and dump the generated matrix. We can then sum up the column values and divide by the



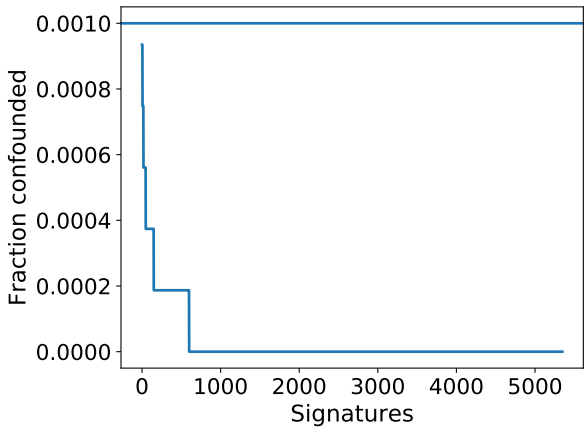
(a) No Distortion

(b) SE Firewall

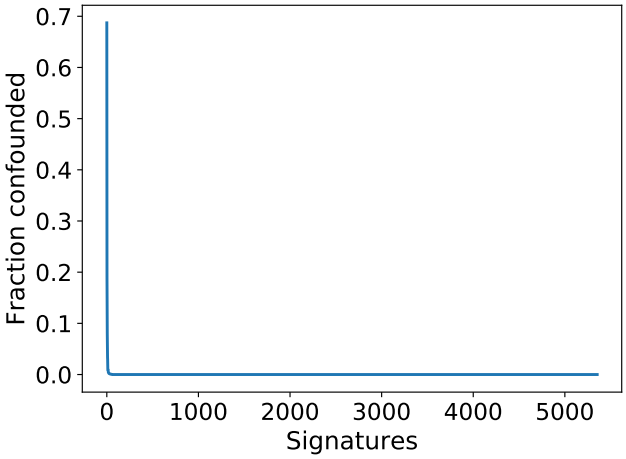
Figure 6.1: Nemesis Reduction Matrix under no distortion and under a SE firewall.

number of signatures in the database to calculate each signatures confounding score. After sorting by descending confounding score we are left with figures 6.2. It is clear that the overwhelming majority of signatures have confounding scores below 0.001. There are, however, a distinct minority of outliers with higher confounding scores.

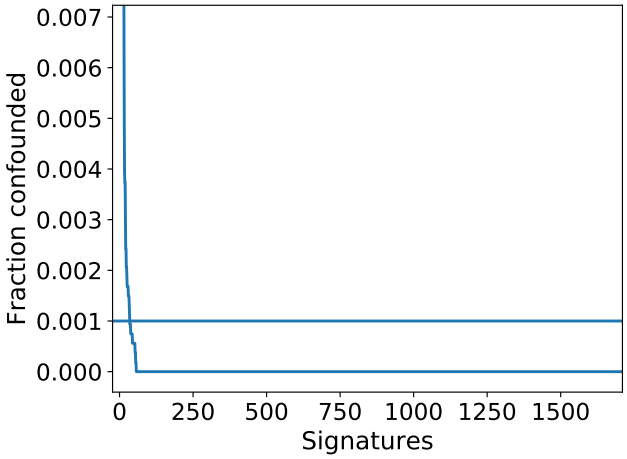
While figures 6.2 are suggestive, they are insufficient when trying to decide on a confounding score maximum threshold. As our true goal is to maximize the number of signatures that can self-match themselves, we can simply run a binary search by either increasing or decreasing the maximum confounding threshold and then following up with a weakness reduction. This, naturally, depends on our assumption that the number of signatures surviving reduction is concave with respect to the confounding threshold. After running said binary search on the S and SE Firewall cases (consult Appendix B), we end up with figures 6.3. Figure 6.3(a) supports our assumption that the number of surviving signatures is concave with respect to the confounding threshold. From figure 6.3(b) we can see that the ideal confound threshold for a database confronted with firewalls that only allow SYN packets through is 0.0225; for a firewall which also allows SYN packets with



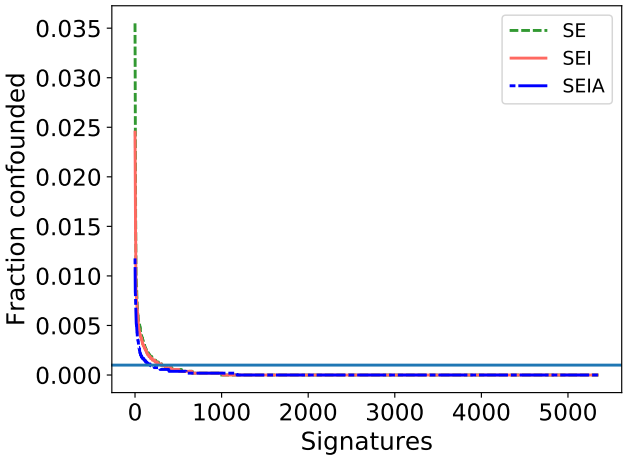
(a) Original Full



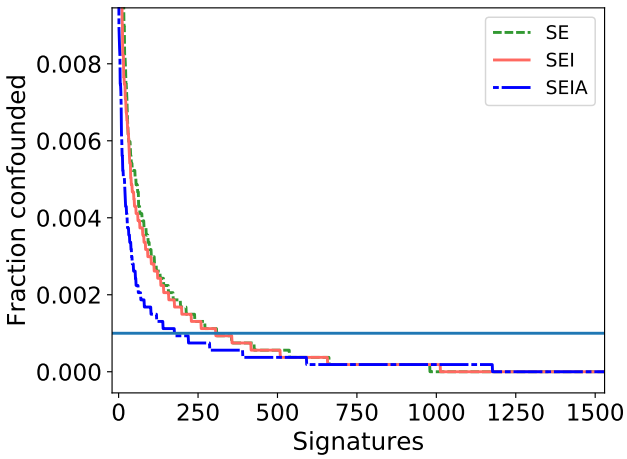
(b) S Full



(c) S Zoomed

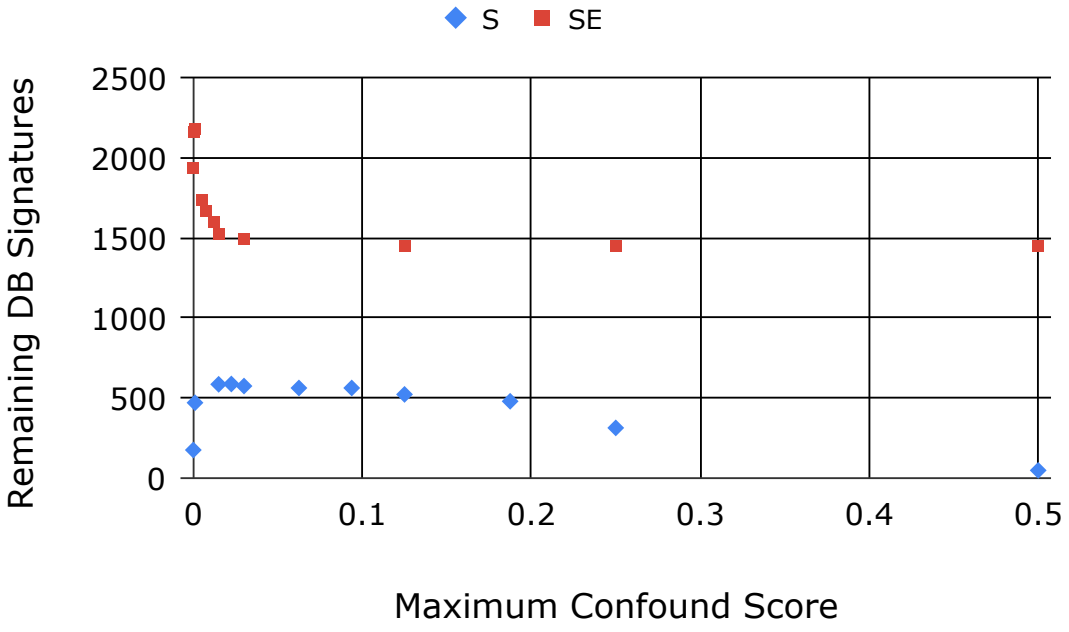


(d) Various Full

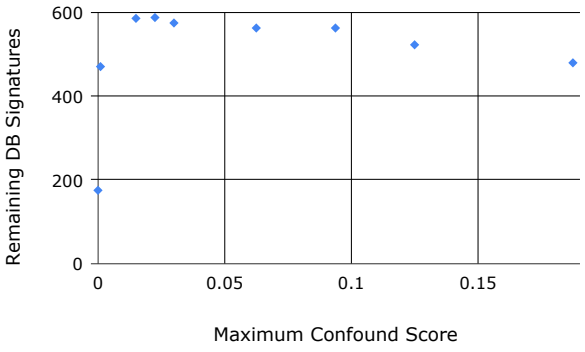


(e) Various Zoomed

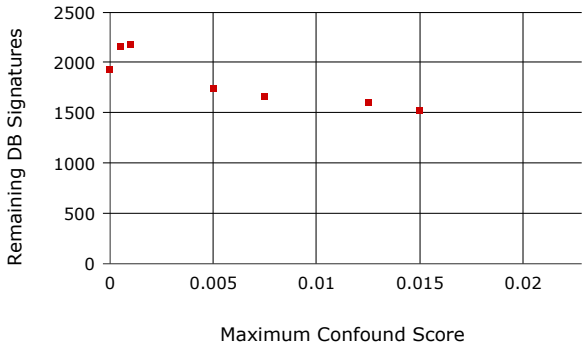
Figure 6.2: Share of signatures confounded by an NMAP DB Signature



(a) Both



(b) S Zoomed



(c) SE Zoomed

Figure 6.3: Amount of NMAP DB Signatures remaining after running a Column (Confound) Reduction, followed by a Row (Weakness) Reduction. Firewalls, either S or SE configuration, were enabled.

Max Confound Score	Remaining DB Values	Net Size
0.001	470	0.087784834
0.0225	587	0.109637654

Table 6.1: Share of signatures surviving a RAC Reduction given a SYN-Only Firewall and given confound scores.

the ECN flag, we see that 0.001 maximizes the number of remaining DB signatures. Comparing with Figure 6.2(d) we can see that this corresponds to the point where the SE database is beginning to flatten out.

While using a search to tailor the maximum confound score to each firewall state does give a benefit, we can see that simply being near 0.001 is generally close enough. For example, in the SYN-only case consider Table 6.1. By adjusting the confound score, we do obtain about 100 more signatures then if we used 0.001. However, this only corresponds to a 2% increase compared to the original size of the database. This is also evident from Figure 6.3(b) and 6.3(c). Though the confound score can have a profound impact, the benefits right around 0.001 are somewhat minimal compared to the difference between 0.001 and, say, 0.5. Thus, given the computational expense of such a search, we'll generally stick to using 0.001 as the maximum confound score unless a confound score matrix dump shows reason to believe the thresholds should be set differently.

Now having determined our thresholds, we can look at the number of confounding signatures actually dropped, as shown in Table 6.2. We can see that in the no Firewall case, there are only 6 signatures that qualify as confounding and must be dropped. As we make the Firewall more restrictive, the number of confounding signatures increases until we get to the most restrictive Firewall state (SYN only, ECN flag forbidden) and the number of confounding signatures begins to drop again. Based on our Firewall scan result in Figure 5.2 it is likely that there are simply few signatures in the database that exhibit a probe response pattern wherein TCP SYN packets return a response unless the have an ECN flag set. And given how important the probe response pattern is to NMAP when determining whether a signature is a match, as show in Section 2.3.2, fewer signatures with the same probe response patterns as the Firewall that the probes are subject

Firewall	Max Confound Score	Signatures Dropped
S	0.0225	31
S	0.001	321
SE	0.001	1042
SEI	0.001	1020
SEIA	0.001	307
N/A	0.001	6

Table 6.2: Number of Confounding Signatures dropped under various Firewall conditions. Consult Appendix B for Firewall configuration explanations.

to implies less potential for signatures to be confounding.

7. THE EFFECT OF FIREWALLS ON DIMENSIONALITY

7.1 Running Reductions

While a Column-Based reduction will remove confounding signatures from the database, we are still left with weak signatures that are unable to match themselves. For this we use Row-Based reduction (see Section 3). Of particular interest is how signatures survive under various Firewall conditions. By looking at Figure 5.2 it is clear that internet firewalls will generally allow through all of the TCP SYN packets, even if the ECN flag is set. Firewalls will also somewhat commonly allow through ICMP packets and TCP ACK packets to closed ports (around 70% of the time). We may thus run database reductions under all of the above firewall conditions (defined in Appendix B).

For this we look towards Table 7.1. From here it is clear that even without any distortion or firewall interference, approximately 10% of the signatures in the NMAP database end up failing to be able to match against themselves reliably and are removed. Even when using one of the least restrictive firewall settings, allowing in all valid TCP packets as well as ICMP packets and a TCP ACK to a closed port results in just over 40% of the database being dropped. It is clear that, in it's default configuration, packet drops carry and enormous impact on the ability of NMAP to properly detect fingerprint Operating Systems within it's database.

We now wish to consider the impact of confounding signatures. That is, signatures that not

Firewall	Row Reduce Direct	Net DB Size
N/A	4872	0.909973851
S	44	0.008218155
SE	1444	0.269704894
SEI	1621	0.302764288
SEIA	3179	0.593761674

Table 7.1: Net size of Database after running a Row-Based (Weakness) reduction under various firewall conditions. Consult Appendix B for Firewall configuration explanations.

Firewall	Row After 0.001 Column	Net DB Size
N/A	4880	0.911468061
S	470	0.087784834
SE	2177	0.406611879
SEI	2289	0.427530818
SEIA	3626	0.677250654

Table 7.2: Net size of Database after running a Column-Based (confound) reduction followed by a Row-Based (Weakness) reduction under various firewall conditions

only match against themselves, but will match against most other signatures. Matching against other signatures prevents those signatures from being able to fully self-match without interference. We first run a Column-Based (confound) reduction (see Section 3) with a maximum confound threshold of 0.001. We choose 0.001 based on Figure 6.2(d). We then run a regular Row-Based (weakness) reduction on the remaining signatures. By comparing Table 7.1 and Table 7.2 it is clear that confounding signatures have a profound effect when a firewall is in place. While there is almost no difference in the number of surviving signatures when no Firewall is in place, 4880 vs 4872 for RAC (Row after Column reduction) vs Row-Only, when a Firewall only allows SYN packets through, the database has more than 10 times as many surviving signatures when using RAC. While the results aren't quite as dramatic for the other cases, they are nonetheless significant. Though, the more permissive the Firewall, the less significant the gains to using RAC. This is likely because NMAP gives an outsize weighting to packet non-arrival. As such, signatures that match the non-arrival pattern of the selected Firewall have a higher likelihood of being confounding.

7.2 Lowering the Weight Given to Probe Non-Responsiveness

The NMAP OS Classifier has a set series of weights associated with each DB feature. In particular, one of said weights involves the receipt of a response to a given NMAP probe. For the TCP SYN probes in particular, the weight associated with receipt or non-receipt of packets is generally more than half of the weights of all the other features associated with that probe combined. This means that a non-response to a probe has an enormous impact on whether or not a signature is capable of matching itself. We see this not only in the example cited in 2.3.2, but

Firewall	Row After 0.001 Column	DB Size Reduction
N/A	4420	0.1744490101
S	3403	0.364400448
SE	3609	0.325924542
SEI	3609	0.325924542
SEIA	3807	0.288942846

Table 7.3: Net size of Database after running a Column-Based (confound) reduction followed by a Row-Based (Weakness) reduction under various firewall conditions with probe response weight (R) set to 20.

also in Table 6.2 where the number of confounding signatures increases dramatically after adding Firewall distortions. This weight decision is curious, especially in light of the fact that which packets an operating system responds to is one of the most configurable features of an Operating System in the form of Firewalls. In particular, on the internet, a system without Firewalls in place is much more the exception than the rule. The use of such a heavy weight, therefore, is almost certainly more of a hindrance than a blessing.

We can, however, investigate the effects of lowering this weight on the NMAP Classifier. To do this, we can rerun the DB reductions under various Firewall conditions with new lower weights for probe non-response. In this instance, we set the probe response weight to 20. For comparison, most features have a weight around 20, whilst the usual probe response weight is set to 100. The results of rerunning this test is detailed in Table 7.3. One thing that is immediately obvious, especially in comparison with Table 7.2, is that there is considerably less of a penalty when the classifier is faced with more restrictive Firewalls. In particular, we see no difference between the cases where a Firewall allows Syn packets (with the ECN flag) and the case where those packets and ICMP packets are allowed through. This implies that, the ICMP packet provides no useful signal when classifying an Operating System save for whether or not said system will respond. Notably, whether or not an OS will respond to an ICMP packet is much more likely to be governed by Firewalls than by OS characteristics.

Also of interest is the effect of the lower weights on the no Firewall case. We can see that in the

no firewall case, 4420 signatures survive when using a low probe response weight. When using the default weights, the no firewall case yields 4880 signatures, as show by Table 7.2. The low R case has a database about 10% smaller, as in this case, the level of responsiveness to probes is a valid signal when trying to identify the operating system in use. This suggests that there is a modest benefit to keeping the old R weights when scanning on, say, a local network where Firewalls are not in play. That being said, given how widely deployed Firewalls are, and how much better NMAP appears to perform in the Firewall case with a lower weight, the current R weight is almost certainly set too high.

8. FEATURE IMPACT ON DIMENSIONALITY

In Section 4.1.3 we briefly discussed the importance of various fields NMAP uses to Fingerprint an OS. Here we will more closely evaluate the importance of those various fields by observing what effect their distortion has on the dimensionality of the NMAP database. Given that we are primarily concerned with OS Fingerprinting over the internet, we will primarily assume that the probes will also be subject to Firewalls, thus restricting the features that make it through.

8.1 Reduction under Distortion

NMAP has a total of 29 unique features which may be distorted. We first begin by deciding which combination of features we wish to distort. We can do this by analyzing the effects of individual distortions on self-match averages on the reduced RAC SYN-only database. When looking at Figure 4.2 it is clear that certain features, in particular, OX (options), TI (TCP IP ID Generation), TS (Timestamp), and WX (TCP Window Size) have an out-sized effect on signature matches. As such, we define distortion profiles in Table 8.1. We can now run a reduction of the database while under various profiles.

We start by taking the already reduced DB under the SYN firewall (when having undergone Column-Based reduction first), and proceed to run a Row-Based reduction on it. A few things are

Distort Profile	Distorted Values
D1	WX, OX
D2	*D1* + DF, T/TG
D3	*D2* + TI
D4	*D2* + GCD, ISR, SP
D5	*D4* + TS, A
D6	*D5* + S, F
D7	*D6* + TI
D8	*D7* + II
ALL	ALL

Table 8.1: Distort Profiles which define which NMAP features will be distorted. Consult Appendix A.1 for brief explanation of features.

Distort Profile	Remaining DB Values	DB Size Reduction
20% D1	463	0.211243612
20% D2	439	0.252129472
20% D2 + jitter_a	371	0.367972743
20% D3	392	0.332197615
20% D4	366	0.37649063
20% D5	285	0.514480409
20% D6	276	0.529812607
20% D7	136	0.768313458
20% All	112	0.809199319
20% All + jitter_a	90	0.846678024
50% D1	454	0.226575809
50% D2	433	0.262350937
50% D4	346	0.410562181
50% D5	223	0.620102215
50% D6	213	0.63713799
50% All	2	0.996592845

Table 8.2: Number of Remaining Signatures in NMAP database after running Row-Based reduction with distortion profile and S Firewall. Column-Based reduction on S Firewall was run first with 0.0225 as confound threshold.

immediately clear from Table 8.2. Firstly, Nemesis does appear to be appropriately distorting the signatures as almost nothing survives the 50% all distortion. Second, it is clear that the features identified in Figure 4.2 as being vital to signature matching are indeed crucial. We see this when we use distortion profile D7, which distorts said features identified in Figure 4.2; the number of remaining signatures in the database is only slightly higher then when all features are distorted. It is also clear that, as more key features are distorted, the number of surviving signatures steadily deceases.

By looking at table 8.3 we can see that our conclusions generalize to the case where SYN packets with the ECN flag set are allowed, despite the fact that the starting database is almost 4 times larger. Just as before, distort profile D7 is almost the same as the all distort profile; we also find a steady decrease in the number of matches as more features are distorted. Indeed, this trend appears to generalize over a wide range of expected Firewall conditions. Looking at Table 8.4 we can see that the number of entries remaining in the database are remarkably similar after using

Distort Profile	Remaining DB Values	DB Size Reduction
20% D1	1777	0.183739091
20% D2	1659	0.237942122
20% D2 + jitter_a	1140	0.476343592
20% D3	1542	0.291685806
20% D4	1475	0.322462104
20% D5	1241	0.429949472
20% D6	1234	0.433164906
20% D7	1130	0.480937069
20% All	1073	0.50711989
20% All + jitter_a	509	0.766192007
50% D1	1782	0.181442352
50% D2	1657	0.238860818
50% D4	680	0.687643546
50% D5	374	0.82820395
50% D6	362	0.833716123
50% All	8	0.996325218

Table 8.3: Number of Remaining Signatures in NMAP database after running Row-Based reduction with distortion profile and SE Firewall. Column-Based reduction on SE only Firewall was run first with 0.001 as the confound threshold.

Profile	SEI	Reduction	SEIA	Reduction	N/A	Reduction
D1	1204	0.474006116	902	0.751241037	1910	0.470767526
All	507	0.778505898	103	0.971594043	105	0.970906068
D7	622	0.728265618	167	0.95394374	263	0.927126628

Table 8.4: Number of Remaining Signatures in NMAP database after running Row-Based reduction with various distortion profiles and Firewall rules. Confound reduction done first with applicable Firewall rules.

distort profiles D7 and All. This strongly suggests that almost all of the useful classification signal comes from the features listed in distort profile D7 amongst all of the firewall configurations. In addition, even in the no Firewall case, it is clear from Table 8.4 the dimensionality of the database in the D7 and All distort case are remarkably similar to the SEIA case. This further confirms that the features distorted in D7 form the core features that provide almost all of the signal.

8.2 Distortion Caused by High Load

It is clear, looking at the dimensionality tests in Section 8.1 that the features identified in Section 4.1.3 are, indeed, vital to correct Fingerprinting of an OS. While it is unlikely all of those features would be distorted by chance on the open internet, like the Firewall distortions, distortions caused by high server load are much more likely. The two main fields that are susceptible to such distortion, as identified in Section 4.2, are the IP ID Generation method and the TCP ISN Feature. We look at the effect of distortion on both fields in the high load case.

In the original IETF standards the IP ID field was required to be unique, though it is now only required to be unique in contexts where IPv4 Packet Fragmentation has or may occur [25]. This means that if IPv4 Packet Fragmentation is permissible the IP ID counter must change between responses. In the context of NMAP, there is an underlying assumption that the IP ID field is not time dependent but number of probes received dependent (that is, it only changes when a new packet arrives) and that the target will not be receiving enough inbound traffic to quickly alter the IP IDs. This assumption does not necessarily hold under high load conditions. Indeed, under sufficiently high load where a server must be constantly changing IP IDs, NMAP will conclude that the server IP ID generation protocol is Random (RD) under the rule, “If the IP ID sequence ever increases by at least 20,000, the value is RD (random).” [19, Chapter 8]

Another anticipated effect of a server under high load is an increase in RTT Jitter which will distort the TCP ISN Features. The reasoning being that a server under high load may not respond to packets immediately, deferring until resources are available. Under a high load scenario, the amount of time an OS will defer responding to a packet can be highly variable, depending on the nature of the other requests and the order in which they were received.

Scale	Shape	DB Size	Reduction on RAC
N/A	N/A	587	0
1	1000000000	522	0.110732538
1	2.59	458	0.219761499
0.5	1.72	448	0.236797274
0.25	1.21	429	0.269165247

Table 8.5: Net size of Database after running a Row-Based (Weakness) reduction under various Jitter Conditions (Scale & Shape define Pareto variable parameters). In all cases, IP ID was distorted to Random (RD).

To test the effects of high load, we decide to look at how the dimensionality of the NMAP Database is effected in the case where an S Firewall is already in place. To do this, we take the already reduced NMAP database, after a column and row reduction, and subject it to distortion. In particular, IP ID is always set to "Random" (RD) and we subject the TCP ISN fields to ever increasing distortion. The result is Table 8.5. We see that there is basically no effect when distorting only the IP ID as the database of 587 items is the same size as the dimensionality of the Database when faced with only an S Firewall. As such, it is clear that the only effect from high load will come from TCP ISN Jitter. Again looking at Table 8.5 we see that under the almost no jitter case, we have a 11% reduction in dimensionality of the database. In the moderate jitter and severe jitter, we have about a 20% and 27% reduction in the dimensionality of the database, respectively. It is clear that network jitter can have a relatively large impact on the dimensionality of the database, even under almost perfect network circumstances. Given the sensitivity of the performance of NMAP to jitter, it makes sense to consider whether or not the ISN parameters are either weighted too heavily or whether the acceptable ranges for the ISN values should be widened to more easily accommodate said network jitter.

9. ZSIGNATURE EMULATOR

While the main thrust of this work is to build a Framework to evaluate OS Fingerprinting Systems using dimensionality, the primary motivation to do so is to assist in the development of tools and methods for reliable internet scale OS Fingerprinting scans. In that vein, zSignature helps lay the foundations for development of future such tools. When zSignature is started up, it reads in an NMAP DB file and loads it into memory. The simulator may be configured to either simulate a single signature, or it may simulate the whole range of NMAP signatures by assigning each simulated signature to an IP address. When NMAP sends a probe to an IP address bound to zSignature, the simulator identifies the incoming NMAP probe and responds as defined by the relevant reference signature. zSignature is quite similar to another tool HoneyD [26]. Though, while HoneyD attempts to simulate a whole network of fake machines, zSignature attempts to accurately emulate a single machine.

The underlying implementation of zSignature is generally quite simple. A custom network driver captures all incoming packets on select IP addresses, the packets are identified as belonging to NMAP or not. If they belong to NMAP, the probe type is identified and appropriate response packets are crafter. Otherwise the packet is dropped. zSignature assumes that all incoming packets are either NMAP probes or invalid packets. With this assumption, it is generally quite easy to identify particular NMAP probes as they are quite distinctive. The first six NMAP TCP probes have differing TCP options and window field values. The T2-T7 probes have various different flags set, whereas the UDP and ICMP probes are distinctive in that they are not TCP packets. While this is clearly not a general approach to simulating machines that can be used with other OS Fingerprinting Systems, this is sufficient for a proof of concept.

The responses to most probes, in particular the T2-T7 probes, are mostly checking for fixed field responses to the probe or are otherwise checking for responsiveness. For reference signature values with a range, the average of the range is returned in the response packet. For values with a list of values separated by a “logical or” the first value is returned. Given how fixed some of

the fixed fields are, proto-packets with most of the features already set can be created from the NMAP Database, with almost all of the remaining features crafted stateless upon the arrival of a probe. Indeed, the few fields that require state to be kept between probes are: Timestamp and Initial Sequence Number Fields. Timestamp requires state as it's value is incremented at some frequency per second, while the Initial Sequence Number Fields specify how fast the ISN is being incremented per second, as well as it's GCD and variance.

The main improvement of zSignature of HoneyD is the way that TCP Initial Sequence Numbers are generated. In particular, HoneyD does not guarantee that it's standard deviation will be within the expected NMAP range over the span of 5 probes. Whereas our implementation of ISN generation, specified in Section 4.2.1, is much more likely to match the expected standard deviation if the jitter is low enough. The reason for this is that HoneyD uses a Random Number Generator to create the standard deviation in it's ISN's. While this should lead to good results over a large amount of probes, it does not guarantee that the standard deviation will be as expected in a fixed number of probes. A more targeted algorithm solves this problem.

A few minor complications cropped up when building zSignature. The first issue is that, to properly craft response packets to NMAP probes, more detail than usual was necessary. In particular, we had to craft packets with our own IP IDs, etc. We also did not wish the system Firewall to drop NMAP probes. Thus, we used a custom networking driver to directly capture the network packets off of the network card before Windows had a chance to really process them. Once we captured our packets and pushed them up into the userspace of zSignature, they were removed from the Windows packet processing pipeline so that Windows would not even be aware of said packets. A similar process was repeated when writing packets. zSignature packets would go directly from userspace to the networking driver, writing it to the network, again bypassing the Windows networking stack. zSignature was given an IP range that it was responsible for and this was how the networking driver determined which packets to filter and which to allow to pass to the Windows networking stack. Packets within the zSignature range were filtered, the rest passed to Windows.

An interesting problem cropped up with this arrangement. zSignature would have one IP ad-

dress per simulated signature. Thus it needed slightly over 5000 IP Addresses. When running under Windows Server 2016, however, it emerged that Windows was not capable of being assigned this many IPs. Once a mere 1000 IPs were assigned the CPU utilization of said server climbed to 100% and stayed there until either the IPs were unassigned or the network card was disabled. Some quick experiments confirmed that as the number of assigned IPs increased, the CPU utilization increased. This was verified on multiple Windows Server 2016 machines. Thus we decided on a workaround. Because packets sent to one of those 5000 IPs would not actually interact with Windows anyway, there was no need for Windows to be aware of those IPs in the first place. The only real reason we needed those IPs to be assigned was to ensure that Windows would transmit the appropriate ARP packets to ensure packets addressed to those IPs were routed correctly. As such, we decided to intercept ARP packets ourselves and capture those asking about IPs within the zSignature range and craft ARP replies ourselves. As such, the Windows networking stack could be kept completely oblivious to all zSignature activities, including the IPs used.

10. CONCLUSIONS

10.1 Conclusions

We have built a general framework for evaluating the effectiveness of OS Fingerprinting Systems and, as a proof of concept, focused on the current de facto OS Fringerprinting standard: NMAP. The primary metric we use when evaluating OS Fingerprinting systems is dimensionality. We also have a secondary metric, number of confounding signatures. In particular, as part of our focus on dimensionality, beofre doing our actual dimensionality calculations we generally first filter out confounding signatures.

For the NMAP OS Fingerprinter, in particular, confounding signatures mainly became a problem when faced with a Firewall. Due to the high weight that probe response carries, a signature that was otherwise valid might better match a different signature whose probe response odds match that of the Firewall; an example of this is shown in Section 2.3.2. We show in Section 6.2 that removing confounding signatures has a highly positive effect on NMAP dimensionality. Even after removing confounding signatures, however, the NMAP database loses about 9% of its signatures even without any distortion and 32% of its signatures when faced with a reasonably lenient Firewall. After lowering the NMAP weight given to probe responsiveness in Section 7.2, we find that the dimensionality of NMAP improves in all cases where a Firewall is present, though performance does degrade in the zero distortion case. Given our results in Section 5 showing the prevalence of Firewalls on the internet, we suggest that NMAP is much more suited for local OS Fingerprinting than internet scale fingerprinting in its default state.

In addition to NMAPs sensitivity to Firewalls, the OS Fingerprinter also shows sensitivity to network jitter. While NMAP generally focuses on fixed fields that would not change, regardless of network conditions or server load, in Section 4.2 we identify specific features that may be impacted by jitter. In particular, IP ID Generation as well as TCP Initial Sequence Number Generation are impacted by network conditions. We describe in more detail the mechanism behind how jitter

can distort those features and describe, in Section 4.2.1, a means of simulating distortion. We show that such jitter can strongly impact the ISN Generation in particular. In Section 8.2 we show that, while IP ID distortion tends not to have any meaningful impact on the dimensionality of NMAP, distortion of the TCP ISN parameters caused by network jitter does have an impact on dimensionality. While this effect is most pronounced under moderate to high network jitter even extremely minor network jitter, like that found on a local network, is enough to hurt the dimensionality of NMAP. This suggests that the way the TCP ISN generation fields are handled may need to be rethought. It also further suggests that NMAP may struggle with internet wide scans.

Finally we looked at which features contain the most signal, that is, which features NMAP looked at were most important to be able to accurately identify an OS. We did this by looking at the effect that distorting various features had on the dimensionality of the NMAP OS Fingerprinter. In Section 4.1.3 we looked at the effect that distorting any individual feature had on the dimensionality of NMAP. We concluded that TCP Options, Window Size, and Timestamp were highly important fields because of their high entropy in the NMAP database. When looking at their individual effect on the ability of any particular signature to match itself, we find that besides the above candidates, TCP IP ID Generation stood out as a very important field. Upon closer consideration, we find that the Windows OS Family, in particular, has a very distinctive way of handling IP ID generation, which may help to explain why it carries so much more singal than its entropy would suggest. Further, when looking at Figure 4.3 we see that the same features are generally important across various Firewall conditions. When looking at the dimensionality of the NMAP database when under the D7 distort condition, that is when the following NMAP fields are distorted [WX, OX, DF, T/TG, TI, GCD, ISR, SP, TS, A, S, F, TI, II], we find that, even in the no Firewall case, that NMAPs dimensionality drops close to zero. Given that these only represent half of the NMAP fields, this suggests that there is considerable scope to reduce the number of probes NMAP sends without sacrificing dimensionality.

Overall, we conclude that the Framework provides a useful metric for evaluating OS Finger-

printing Systems under various conditions. We conclude that NMAP, the de facto OS Fingerprinting standard, in particular is ill suited to take on internet wide OS scans. This suggests that considerable future work is necessary to either update NMAP to meet such a goal or that a new tool must be built from scratch.

10.2 Further Study

Given our framework, we can explore more deeply several interesting questions in the future. For example, we can more deeply study how many probes and tests can be eliminated without sacrificing dimensionality. There is also great potential to study the weights NMAP assigns to each feature. For instance, we only looked at the effect of reducing the weight of probe non-response. Future works could develop a system which maximizes the dimensionality given the set of weights NMAP has. Further effort could be deployed into seeing what effect Firewalls have on ideal NMAP weights.

In addition, Nemesis DB Reduction could be used on other systems facing similar parameters as NMAP did here in an effort to provide a head to head comparison of the dimensionality of NMAP and said OS Fingerprinting systems like, for example, Plata [18]. zSignature could be expanded upon into a more general tool that can be used to train new OS Fingerprinting systems based on NMAP signatures. Alternatively, it's advancements over HoneyD, like the TCP ISN Signature generator, could be incorporated into HoneyD and the newly advanced system could be used to train new Fingerprinting systems.

REFERENCES

[1] C. Dwyer, "Hackers accessed the personal data of 143 million people, equifax says," Sep 2017.

[2] F. Langfitt, "British hospitals among targets of global ransomware attack," May 2017.

[3] J. Sutherland, "Millions of us computers completely pwned by botnets.," Oct 2010.

[4] J. Griffiths, "Japan is hacking millions of its citizens because everyone's home security is terrible," Feb 2019.

[5] "Ettercap home page." `http://www.ettercap-project.org/`.

[6] "p0f v3 (version 3.09b)." `http://lcamtuf.coredump.cx/p0f3/`.

[7] "Nmap: The network mapper." `https://nmap.org/`.

[8] R. R. Rohrmann, V. J. Ercolani, and M. W. Patton, "Large scale port scanning through tor using parallel nmap scans to scan large portions of the ipv4 range," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 185–187, July 2017.

[9] R. Jicha, M. W. Patton, and H. Chen, "Identifying devices across the ipv4 address space," in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pp. 199–201, Sept 2016.

[10] G. Shu and D. Lee, "A formal methodology for network protocol fingerprinting," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 1813–1825, Nov 2011.

[11] T. Matsunaka, A. Yamada, and A. Kubota, "Passive os fingerprinting by dns traffic analysis," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 243–250, March 2013.

[12] J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Siphon: Towards scalable high-interaction physical honeypots," in *Proceedings*

of the 3rd ACM Workshop on Cyber-Physical System Security, CPSS '17, (New York, NY, USA), pp. 57–68, ACM, 2017.

[13] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *CoRR*, vol. abs/1608.06249, 2016.

[14] D. Hanna, P. Veeraraghavan, and B. Soh, "Sdmw: Secure dynamic middleware for defeating port and os scanning," *Future Internet*, vol. 9, no. 4, 2017.

[15] D. Watson, M. Smart, G. R. Malan, and F. Jahanian, "Protocol scrubbing: network security through transparent flow modification," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 261–273, April 2004.

[16] "Nmap: Tcp/ip fingerprinting methods." `https://nmap.org/book/osdetect-methods.html`.

[17] J. Postel, "INTERNET CONTROL MESSAGE PROTOCOL," RFC 792, Internet Engineering Task Force, September 1981.

[18] Z. Shamsi and D. Loguinov, "Unsupervised clustering under temporal feature volatility in network stack fingerprinting," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 2430–2443, Aug. 2017.

[19] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

[20] H. Dahmouni, A. Girard, and B. Sansò, "An analytical model for jitter in ip networks," *annals of telecommunications - annales des télécommunications*, vol. 67, pp. 81–90, Feb 2012.

[21] E. J. Daniel, C. M. White, and K. A. Teague, "An interarrival delay jitter model using multi-structure network delay characteristics for packet networks," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, pp. 1738–1742 Vol.2, Nov 2003.

[22] L. Rizo-Dominguez, D. Torres-Roman, D. Munoz-Rodriguez, and C. Vargas-Rosales, "Jitter in ip networks: a cauchy approach," *IEEE Communications Letters*, vol. 14, pp. 190–192, February 2010.

[23] G. Hooghiemstra and P. V. Mieghem, "Delay distributions on fixed internet paths," 2002.

[24] "University of oregon route views archive project." `http://routeviews.org/`.

[25] J. Touch, "Updated Specification of the IPv4 ID Field," RFC 6864, Internet Engineering Task Force, February 2013.

[26] N. Provos, "A virtual honeypot framework," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, (Berkeley, CA, USA), pp. 1–1, USENIX Association, 2004.

APPENDIX A

NMAP DATABASE MINUTIA

A.1 NMAP Brief Feature Description

Below is a list of all the various NMAP features and a brief description of their meaning. A more complete description is available at [16].

- A: Acknowledgment number tests, is it same as SEQ in probe? Zero? Other?
- CC: Explicit congestion is supported
- CD: Is IMCP Response Code 0 (correct) or echoed back from incorrect ICMP request
- CI: IP ID Sequence Generation Alogrithm based on responses from TCP requests to a closed port.
- DF: Don't Fragment bit, is it set?
- DFI: Don't Fragment bit for ICMP, is it set? (N: no, S: echo DF of probe, Y: yes)
- F: TCP Flags in response
- GCD: TCP Initial Sequence Number greatest common divisor.
- II: IP ID Sequence Generation Algorithm based on responses from ICMP requests.
- IPL: Total length of ICMP Unreachable Response to UDP probe.
- ISR: TCP Initial Sequence Number counter rate. How quickly does ISN tick up?
- O: TCP Options (special note, this is almost always blank for any of the non-standard normal TCP probes, usually only set for good TCP SYN or TCP SYN-ECN)
- Q: Special quicks in TCP stack
- R: Is a response expected to this probe?
- RD: Checksum of ASCII error message in RST packet. 0 is no such data

- RID: In the ICMP Port Unreachable message (for UDP probe), is the IPID of the returned UDP header unchanged?
- RIPL: Returned ICMP probe IP Total Length Value, is the value of the ORIGINAL IP Header correct (returned in the ICMP Port Unreachable message)
- RUCK: In the ICMP Port Unreachable message (for UDP probe), is the UDP checksum of the header unchanged?
- RUD: Integrity of the returned UDP data in the ICMP Port Unreachable Message. This shouldn't have been mangled.
- S: TCP Sequence Number test in relation to TCP Ack number from the probe that elicited the response
- SP: TCP Initial Sequence Number variance.
- SS: Shared IP ID, is the IP ID Sequence shared between TCP and ICMP protocols?
- T: IP TTL value
- TG: IP TTL value "guess" when exact hop distance not known (occurs when no ICMP Port Unreachable message comes from UDP probe)
- TI: IP ID Sequence Generation Algorithm, based on original 6 TCP SEQ probes.
- TS: TCP timestamp option algorithm
- UN: Unused port unreachable field nonzero (RFC mandates it should be zero)
- W: Initital window size

A.2 The Confounding Tomato

Reproduced below from the NMAP database is an example of the Tomato signature, which tends to be extremely confounding for NMAP when NMAP is scanning targets behind firewalls:

Linux 2.6.18-4-ixp4xx #1 Sun Apr 22 08:34:11 UTC 2007 armv5tel GNU/Linux, Debian on

NSLU2

Tomato USB Version 1.28 Tomato Firmware v1.28.7821 MIPSR1-Toastman-ND K26 MiniIPv6

- Linux kernel 2.6.22.19 and Broadcom Wireless Driver 5.10.147.0 updates

Fingerprint (2457) Linux 2.6.18 - 2.6.22

Class Linux | Linux | 2.6.X | general purpose

CPE cpe:/o:linux:linux_kernel:2.6 auto

SEQ(R=N)

OPS(R=N)

WIN(R=N)

ECN(R=N)

T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0%Q=)

T2(R=N)

T3(R=N)

T4(R=N)

T5(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)

T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)

T7(R=N)

U1(DF=N%T=3B-45%TG=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)

IE(DFI=N%T=3B-45%TG=40%CD=S)

A.3 A Windows Signature

Reproduced below from the NMAP database is an example of a signature for the “Windows 10” Operating System:

Windows 10, DC=D

Windows 10 1607

Fingerprint (3398) Microsoft Windows 10

Class Microsoft | Windows | 10 | general purpose

CPE cpe:/o:microsoft:windows_10 auto

SEQ(SP=FC-106%GCD=1-6%ISR=108-112%TI=I%CI=I%II=I%SS=S%TS=A)

OPS(O1=M4ECNW8ST11%O2=M4ECNW8ST11%O3=M4ECNW8NNT11%O4=M4ECNW8ST11

%O5=M4ECNW8ST11%O6=M4ECST11)

WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)

ECN(R=Y%DF=Y%T=7B-85%TG=80%W=2000%O=M4ECNW8NNS%CC=N%Q=)

T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=S+%F=AS%RD=0%Q=)

T2(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)

T3(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)

T4(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)

T5(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)

T6(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)

T7(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)

U1(DF=N%T=7B-85%TG=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)

IE(DFI=N%T=7B-85%TG=80%CD=Z)

APPENDIX B

FIREWALL CONFIGURATION EXPLANATIONS

The following are the various Firewall configurations used throughout this Thesis.

- S: A Firewall that only allows valid TCP packets with the SYN flag set. All other probes are dropped, including valid TCP packets with the SYN and ECN flag set. What this means for NMAP, in particular, is that only the T1 and SEQ tests can be completed as they are the only tests that use valid TCP packets with the SYN flag set.

- SE: The same as S except that valid TCP SYN packets with the ECN flag set are also allowed through. This allows for the NMAP ECN tests to also be completed. It is clear from Figure 5.2 that valid TCP SYN packets with the ECN flag set are rarely blocked. Thus, this is almost certainly the most common firewall configuration.

- SEI: The same as SE except that ICMP packets are now also allowed through the Firewall. This includes the valid ICMP Echo requests as well as the (lightly) malformed ICMP Echo request; an ICMP ECHO request must set the CODE field to 0, but because this field has no other possible values in an ICMP ECHO request, few Operating Systems check this field. The IE NMAP feature requires both ICMP packets to return for the ICMP tests to return features. Further, we can see from Figure 5.2 that few Firewalls allow valid ICMP Echo requests but block invalid ICMP Echo requests.

- SEIA: The same as SEI except that valid TCP packets with the ACK flag set sent to a closed port are allowed through the Firewall. In this case, the expected response would be a TCP packet with the RST flag set. This allows for the T4 test to be completed.

- N/A or ALL: All packets are allowed through the Firewall. That is, there if no Firewall in place and all probes make it through allowing all NMAP tests to be conducted.

APPENDIX C

NMAP ISN FIELDS

Given a list of $isnDelta$'s and corresponding $isnTimeDelta$'s, we define the following. Note that seq_rates and seq_rates_2 are intermediate values defined by us to more easily calculate ISR and SP, which are NMAP fields.

1. GCD: Simply the Greatest Common Denominator between all of the $isnDelta$'s.
2. seq_rates: Divide each $isnDelta_i$ by it's corresponding $isnTimeDelta_i$ to get the seq_rates array.
3. ISR: This is simply $8 * log_2(mean(seq\_rates))$ rounded to the nearest integer.
4. seq_rates_2:

$$\begin{cases} \text{Divide each element in } seq\_rates \text{ by } GCD & GCD > 9 \\ \text{The same as } seq\_rates & otherwise \end{cases}$$

5. SP: This is simply $8 * log_2(std\_dev(seq\_rates_2))$ rounded to the nearest integer (where std_dev is the standard deviation).